Fabien Zennaro
EPFL
Informatique 4ème année

LSR
Assistant : Matthias Wiesmann
Février 2004

Mesure de performance d'un service de détection de pannes SNMP

1. Introduction

Le but de ce projet est d'expérimenter la détection de panne sur un réseau au moyen d'un équipement compatible avec le protocole SNMP. L'idée est d'avoir un appareil tel qu'un switch ou un routeur possédant certaines capacités de monitoring, et d'utiliser celles-ci afin d'informer le réseau entier sur les pannes qui se produisent. Le choix du protocole SNMP est justifié d'une part par la simplicité de ce protocole, d'autre part par le fait qu'il est plutôt répandu dans les équipements susceptibles de convenir à ce type d'expérimentation.

Dans un premier temps, il va falloir déterminer la faisabilité d'un tel système de détection, définir dans quelles conditions il est applicable, et finalement, mesurer les capacités et les performances d'un tel système.

Pour mener mon expérience, j'ai procédé de la manière suivante :
Pour commencer, j'ai effectué des recherches sur les équipements réseaux susceptibles d'offrir de telles capacités de détection. Ensuite, avec peine mais non sans succès, j'ai pu obtenir du matériel pour effectuer mes expérimentations. J'ai également mis au point un petit programme en Java me permettant de récupérer les informations transmises par l'équipement. Pour finir, j'ai étudié plusieurs cas de pannes réseau fréquentes, et j'ai mis au point une série d'expériences afin de me rendre compte des capacités d'un tel système de détection.

Dans la suite de mon rapport, je vais détailler une à une chacune des étapes de mon travail, en terminant par des exemples d'application d'un tel système de détection, ainsi que des informations sur les équipements utilisables pour ce système.

2.1 Détection de pannes

Le principe de base de la détection de panne que je considère dans ce projet est d'utiliser des informations comme le trafic ou l'état d'un port afin d'obtenir des informations sur une partie d'un réseau. Pour obtenir ce type d'information, l'idée mise en œuvre est de les récupérer grâce à un équipement compatible avec le protocole SNMP.

2.1 Type de pannes

Nous avons en réalité deux types d'information sur l'état d'une machine. D'une part le trafic réseau qui peut nous informer sur l'état de la machine, d'autre part l'état du lien, qui nous informe plutôt sur la connectivité du

réseau. Je me suis rapidement rendu compte que les informations sur le trafic étaient difficilement exploitables car une absence de trafic n'est pas obligatoirement liée à une panne. Il faut donc dans ce cas savoir si l'on attend une information ou pas. Je me suis donc concentré sur les informations physiques, à savoir l'état du lien.

Dans ce cas, nous avons plusieurs types de pannes relativement fréquentes. Dans beaucoup de cas, le lien est tous simplement absent. Ceci peut être soit dû à l'absence de connectivité (câble débranché) soit dû au fait que la machine que l'on monitore n'est pas physiquement disponible (machine éteinte ou carte réseau endommagée). Dans d'autre cas, les problèmes peuvent venir d'un câble réseau défectueux, dont certains brins seraient mal connectés.

Une fois ces observations effectuées, on se rend compte que les symptômes des pannes peuvent être observés sur deux niveaux :

Au niveau connectivité, l'équipement ne détecte pas de lien
Au niveau utilisation, les données ne transitent pas, ou elles sont corrompues.

C'est pour cette raison que j'ai créé une série d'expériences afin de déterminer pour chacune des pannes, les symptômes détectés par l'équipement.

2.2 Détection des pannes

Une fois les types de pannes connus, je me suis demandé de quelle manière un équipement pourrait les détecter. J'en ai alors conclu qu'il y avait principalement deux manières envisageables, dépendantes du type de panne à détecter.

- **Panne au niveau connectivité :** L'équipement transmet les informations au niveau de la connexion des câbles, de manière semblable au leds présentes sur le panneau d'affichage.

- **Panne au niveau transmission :** dans ce cas l'équipement doit analyser les données qui transitent par le câble.

Les expériences que j'ai mises au point devraient me permettre de déterminer le type d'analyses effectuées par l'équipement.

2.3 SNMP

Le protocole SNMP [1] est un protocole très simple basée sur l'interrogation d'une base d'information, développé pour l'administration de réseau. L'idée générale est d'obtenir des informations sur une machine, en demandant la valeur d'une entrée de la base de données de celle-ci. Cette base de données est appelée MIB (Managment Information Base). Chaque appareil compatible SNMP possède sa propre MIB. Celle-ci est organisée de manière hiérarchique à la manière d'un arbre.

Dans le contexte d'une administration par SNMP, nous avons toujours au minimum deux machines en cause. L'une d'elle est appelée le « moniteur », c'est la machine qui souhaite recevoir des informations. La machine dont on veut obtenir des informations s'appelle l'agent.

Afin d'accéder à un élément de cet MIB, une nomenclature a été mise au point. Chaque nœud de l'arbre de la MIB est identifié par un numéro. Une MIB standard a été définie afin de permettre une interopérabilité des systèmes. En voici un extrait :

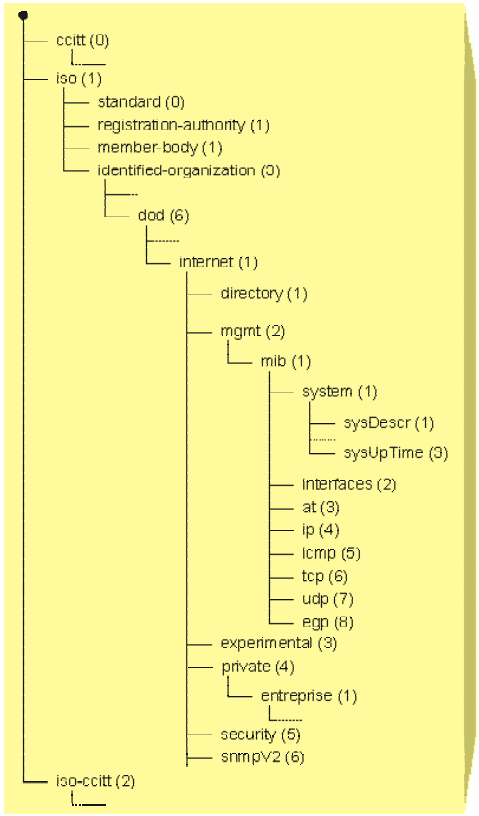

Figure 1 – hiérarchie de la MIB standard

Pour accéder à une entrée sui se trouverait dans la branche « interfaces » il faudrait utiliser la nomenclature : 1.3.6.1.2.1.2. Cette suite de numéros séparés par des points est appelée OID (Objet Identifier) de l'entrée de la MIB.

Je vais m'intéresser à deux principales catégories d'entrées. Les entrées standard et les entrées spécifiques à une entreprise. En effet, les implémentations de SNMP varient selon les constructeurs, et nous avons un mélange d'entrées standard et d'entrées définies par des entreprises.

Pour réaliser l'interrogation de cette MIB, nous pouvons utiliser deux principes. Soit le moniteur envoie des requêtes SNMP pour obtenir une information bien précise (pooling), soit il reçoit une trappe de la part de l'agent. Une trappe correspond en fait à une information non sollicitée, qui est envoyée à la suite d'un événement sur l'agent. Une trappe est donc un message tout à fait asynchrone.

2.3.1 Trappes VS polling

La plupart des documents parlant de SNMP privilégient la communication par trappes. En effet, son caractère asynchrone évite un gaspillage de trafic inévitable en mode polling, car dans ce cas, il n'y à qu'un seul message qui circule. D'autre part, la plupart des réseaux comportent une grande quantité d'agents, qui eux-mêmes proposent un grand nombre d'entrées dans leur MIB. Dans le cas du polling, il devient très lourd de surveiller un très grand nombre d'entrées.

Cependant, le mode par polling ne peut être complètement éliminé. Il reste indispensable pour des applications telle que la découverte de topologie ou la recherche ponctuelle d'information.

Dans notre cas, la communication par trappes est évidemment la plus efficace. D'autre part, la plupart des équipements réseaux implémentent cette fonctionnalité pour les raisons évoquées plus haut.

2.3.2 Détection de panne avec SNMP

L'idée consiste maintenant à utiliser les possibilités du protocole afin de détecter des pannes pouvant intervenir sur un réseau. Trois possibilités s'offrent alors à nous :

Envoie de trappes à intervalle régulier par la station monitorée :

C'est le principe mis en œuvre par Fabien Reichenbach [2]. Dans ce cas, le moniteur attend une trappe à intervalle régulier de la part de la station monitorée. S'il ne la reçoit pas en temps voulu, il peut alors suspecter une défaillance de cette station.

Avantages :
- On peut détecter un très grand nombre de pannes, notamment un niveau logiciel.
- Souplesse d'implémentation.
- Ne nécessite pas d'appareil supplémentaire

Inconvénient :
Il est impossible d'avoir la certitude qu'une défaillance s'est produite.

Interrogation de la MIB d'un équipement réseau.

Avantages :
- Assurance de l'authenticité des pannes
- Informations précises sur la nature de la panne

Inconvénient :
- Très dépendant des capacités de l'équipement
- Beaucoup de trafic

Interception de trappes SNMP venant d'un équipement réseau.

Avantages :
- Assurance de l'authenticité des pannes.
- Informations précises sur la nature de la panne.
- Rapidité de l'information.

Inconvénients :
- Très dépendant des capacités de l'équipement
- Très souvent, qu'une seule partie de la MIB est disponible

Il faut préciser que dans le cadre de mon projet, seules les 2 dernières possibilités sont applicables, dans la mesure où la première ne nécessite pas d'équipement réseau.

Pour mon projet, j'ai décidé d'étudier la troisième des possibilités, car son côté asynchrone permet une détection à la fois fiable et rapide. Cependant, toute son efficacité dépendra de l'implémentation du matériel.

La principale difficulté consiste en l'étude des capacités du matériel du marché. Il faut donc dans un premier temps effectuer des recherches afin de déterminer si oui ou non les appareils du marché sont capables d'envoyer des trappes, et le cas échéant, déterminer quelle type de trappe ces équipements sont capables de générer.

2.4 Principe de détection mis en œuvre.

Comme décrit précédemment, l'idée est d'intercepter des trappes SNMP générées par un équipement placer dans un réseau.

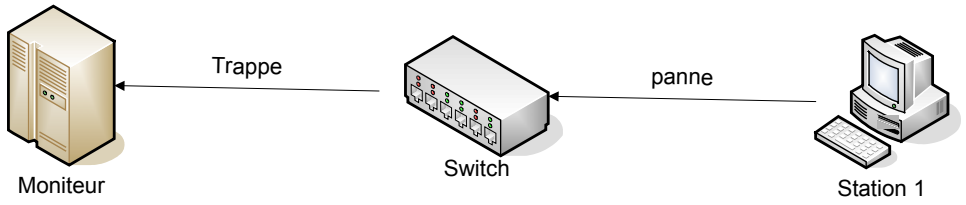

Figure 2.1 – principe de détection simple

Afin d'augmenter l'efficacité du système dans un réseau, il serait intéressant que chaque machine monitorée puisse traiter directement les informations envoyer par l'équipement utilisé pour la détection :

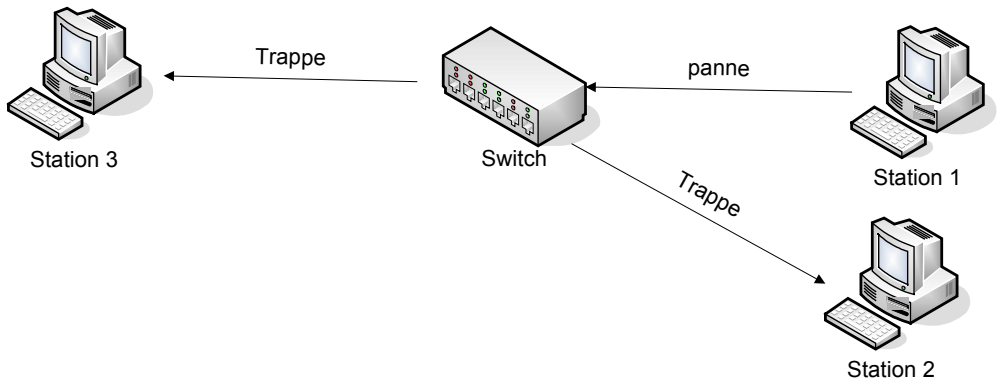

Figure 2.2 – principe de détection avec notification répartie

Dans ce cas de figure, chaque machine du réseau est continuellement informée et peut traiter les pannes qui sont intervenues.

3. Equipement SNMP

3.1 Recherche de matériel compatible SNMP :

Caractéristique de matériel recherché :

- Compatible avec SNMP
- Envoie de trappes utilisables pour détecter des pannes
- Envoie de trappes à plusieurs machines simultanément.

La première étape de mon projet est de trouver du matériel réseau administrable par SNMP, qui contient des informations suffisantes pour réaliser de la détection de panne. J'ai d'abord utilisé du matériel bon marché que j'avais à ma disposition mais je me suis rendu compte après pas mal d'essais qu'il fallait quand même un matériel semi-professionnel pour réaliser ce type d'expérience. J'ai donc décidé de prendre contact avec la société Cisco afin d'utiliser leur matériel et de me renseigner sur ses capacités.
Cependant, mes tentatives sont restées veines, et l'on m'a conseillé de prendre contact avec le service informatique de l'EPFL. Grâce à leur coopération, j'ai pu obtenir du matériel professionnel, ayant une implémentation de SNMP. Il s'agit d'un switch haut de gamme de CISCO.

3.2 Caractéristiques des équipements compatibles SNMP.

Pour qu'un équipement puisse jouer le rôle d'agent, il faut tout d'abord qu'il possède une interface d'administration. Dans un premier temps, je pensais que cette interface était dissociée du reste de l'appareil, et qu'elle ne pouvait être utilisée que de cette manière.

En étudiant le matériel que j'avais à disposition, je me suis vite rendu compte qu'il possédait un port particulier, mais celui-ci est en fait un port série utilisé pour l'administration de l'équipement et non pour le monitorer. En parcourant ensuite la documentation de l'équipement, il s'est avéré que l'on peut définir une interface d'administration virtuelle, que l'on peut associer à n'import quel port de l'équipement. D'autre part, on peut également lui associer une adresse IP, élément qui me paraissait indispensable pour une telle application.

4. Expérimentations

4.1 Introduction

Le but de cette phase de mon projet est de déterminer quels types de pannes sont détectables, et de quelle manière ceci est possible. Afin de répondre à ces questions, j'ai mis au point une série d'expériences. Pour définir ces expériences, je me suis basé sur les principales causes de panne qui peuvent intervenir sur un réseau. J'ai également cherché à créer des expériences qui pourraient me permettre de savoir si l'équipement se contente d'effectuer des contrôles au niveau physique (état du lien) ou s'il analyse les données transitant par ses ports.

4.2 Présentation du matériel

L'équipement que j'avais à disposition est un CATALYSTE 2950 de chez CISCO. Il s'agit d'un switch 48 ports. C'est un modèle professionnel conseillé pour des grands réseaux. Il possède énormément de fonctionnalités, notamment la possibilité de gérer des clusters. Le point important est qu'il possède une implémentation de SNMP, et notamment la possibilité d'envoyer des trappes à la suite d'événements choisis. Selon CISCO [3], leur implémentation de SNMP permet également d'envoyer des trappes à plusieurs machines de manière simultanée, ce qui est également un point intéressant pour la détection que nous voulons mettre en œuvre.

4.3 Premier tests.

Les premiers contacts avec l'équipement ont été assez laborieux. En effet, ce type de matériel par ses capacités est également passablement compliqué à prendre en main.

Dans un premier temps, j'ai du réalisé la configuration initiale du switch en utilisant la liaison série. Lors de cette phase, j'ai du notamment définir les codes d'accès pour l'administration. J'ai ensuite pu me lancer dans la configuration de l'interface virtuelle d'administration dont j'ai parlé plus haut. Pour ce faire, j'ai choisi un port du switch auquel j'ai également attribué une adresse IP (192.168.1.1). L'étape suivante a été de configurer la gestion de SNMP.

Pour réaliser cette étape, on peut procéder de deux manières. Soit on configure SNMP via l'interface série, soit on peut passer par une administration basée sur le WEB. C'est cette dernière que j'ai choisie pour

des raisons de commodités. Cette interface d'administration est basée sur une applet JAVA appelée « Cluster Management Suite Software ». Elle permet notamment de configurer certains paramètres du switch, dont SNMP.

4.4 Configuration de SNMP.

Pour la configuration de SNMP, nous avons plusieurs données à renseigner. Tout d'abord, il faut activer la gestion des trappes. Une fois ceci fait, on doit indiquer la communauté, « public » dans notre cas, le port pour la réception de trappes (8888) et l'adresse du moniteur. Ceci créer une configuration. Afin d'informer plusieurs moniteurs, ce qui est recherché dans notre cas, il suffit de créer plusieurs configurations en précisant les adresses IP des moniteurs souhaités.

Une fois l'étape de configuration terminée, j'ai procédé à quelques testes simple afin de confirmer l'exactitude de mes paramètres. Pour faire ces premiers tests, j'ai utilisé un sniffer (Ethereal) afin d'observer les paquets UDP qui transitaient sur le réseau. J'ai également simulé une panne en débranchant un des ports du switch. J'ai pu observer qu'il y avait effectivement des trappes SNMP qui arrivait en provenance du switch.

Les expériences

Afin d'effectuer mes expériences, j'ai préparé un montage composé de trois machines et du switch. Dans cette configuration, deux machines jouent le rôle de moniteur, et la troisième est utilisée pour simuler les pannes. Le fait d'avoir trois machines m'a permis de vérifier tout au long de mes expériences que l'information d'une panne pouvait être transmise à plusieurs machines en simultané.

Lors de la configuration initiale du switch, j'ai choisi d'installer l'interface d'administration sur le port numéro 1 du switch. J'ai donc utilisé d'une part la machine 1 pour configurer SNMP, d'autre part comme moniteur.

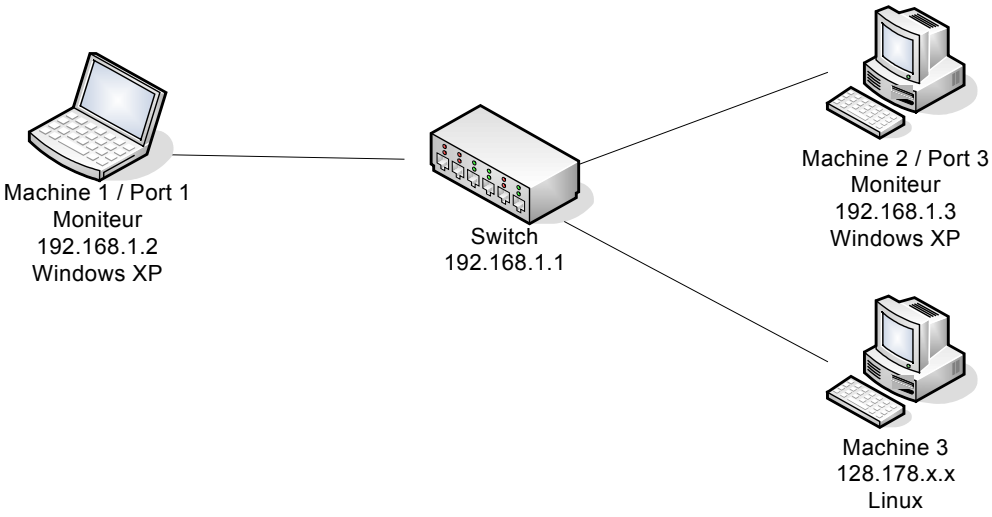

Figure 4.1 – Schéma du montage

Les 2 machines réalisant la fonction de moniteur tournent respectivement sur Windows XP et Windows 98. Le choix de Windows 98 a été dicté pour la facilité de simulé des « écrans plus » sur cette version. La troisième machine tourne sous Linux. Les interconnections entre les machines et le switch ont été réalisée à l'aide de câbles Ethernet droits.

Les adresses IP des machines se trouvent toutes dans le même sous réseau (192.168.1.x), hormis celle de la station sous Linux. Ceci m'a permit notamment de vérifier que le switch ne tenait pas compte des adresses IP des machines pour effectuer la détection de panne.

Pour l'analyse des trappes transmises sur le réseau, j'ai créé un petit programme Java à l'aide du Framework de Fabien Reichenbach. Ce programme intercepte les trappes venant d'un certain hôte et en indique le contenu. Pour chacune de mes expériences, je montrerai la trace que j'ai relevée, en précisant à chaque fois les paramètres importants de chaque trappe reçue. L'analyse et l'explication des trappes seront effectuées dans la partie discutant des résultats.

4.5 Expérience n°1 : débranchement du câble réseau

But : le but de cette expérience est de simuler ce qui se passe lors de l'arrêt d'une machine ou du débranchement d'un câble. Ceci est simplement fait en débranchent le câble du switch.

Etat initial : Les trois machines sont actives et connectées au switch.

Déroulement : à un moment donné, je débranche le câble qui provient de la machine 3.

Etat final : Seul les machines 1 et 2 sont encore connectées au switch.

Trappes reçues :

Trappe n° 1 :

```
host=192.168.1.1,
port=8888,
community=public
enterprise=1.3.6.1.4.1.9.1.429,
generic_trap=Link Down,
Varbinds=[1.3.6.1.2.1.2.2.1.1.5:5,
1.3.6.1.2.1.2.2.1.2.5:FastEthernet0/5,
1.3.6.1.2.1.2.2.1.3.5: 6,
1.3.6.1.4.1.9.2.2.1.1.20.5: down, ]
```

Trappe n° 2

```
host=192.168.1.1,
port=8888,
community=public],
enterprise=1.3.6.1.4.1.9.9.41.2,
generic_trap=Enterprise,
Varbinds=[1.3.6.1.4.1.9.9.41.1.2.3.1.2.14:LINK,
1.3.6.1.4.1.9.9.41.1.2.3.1.3.14:4,
1.3.6.1.4.1.9.9.41.1.2.3.1.4.14:UPDOWN,
1.3.6.1.4.1.9.9.41.1.2.3.1.5.14:Interface    FastEthernet0/5,
changed state to down,
1.3.6.1.4.1.9.9.41.1.2.3.1.6.14: 411606, ]
```

Observations :
Le débranchement d'un câble est bel et bien détecté par l'équipement. On obtient deux trappes ayant visiblement la même signification. Une des trappes est générique, l'autre est spécifique à CISCO.

4.6 Expérience n°2 : débranchement du câble réseau pendant un transfert de fichier.

But : Le but de cette expérience est de simuler la coupure d'un lien lors d'un transfert de fichier. L'idée est d'observé si l'équipement à une réaction complémentaire en détectant la coupure d'une transmission.

Etat initial : Les trois machines sont actives et connectées au switch. Un transfert de fichier est en cours entre la machine 1 et la machine 2.

Déroulement : pendant le transfert, je débranche le câble qui provient de la machine deux.

Etat final : Seul les machines 1 et 3 sont encore connectées au switch. Le transfert de fichier est interrompu.

Trappes reçues :

Trappe n° 1 :

```
host=192.168.1.1,
port=8888,
community=public,
enterprise=1.3.6.1.4.1.9.1.429,
generic_trap=Link Down,
Varbinds=[1.3.6.1.2.1.2.2.1.1.3:3,
1.3.6.1.2.1.2.2.1.2.3:FastEthernet0/3,
1.3.6.1.2.1.2.2.1.3.3: 6,
1.3.6.1.4.1.9.2.2.1.1.20.3: down, ]
```

Trappe n° 2

```
host=192.168.1.1,
port=8888,
community=public,
enterprise=1.3.6.1.4.1.9.9.41.2,
generic_trap=Enterprise Specific,
Varbinds=[1.3.6.1.4.1.9.9.41.1.2.3.1.2.24:LINK,
1.3.6.1.4.1.9.9.41.1.2.3.1.3.24:4,
1.3.6.1.4.1.9.9.41.1.2.3.1.4.24:UPDOWN,
1.3.6.1.4.1.9.9.41.1.2.3.1.5.24:nterface     FastEthernet0/3,
changed state to down,
1.3.6.1.4.1.9.9.41.1.2.3.1.6.24: 675263, ]
```

Observations :
Le comportement est exactement le même que lors d'une coupure sans communication à la différence près que ce n'est pas la même machine qui est concernée. Il n'y a aucune indication supplémentaire.

4.7 Expérience n°3 : branchement d'une machine sur le réseau

But : Le but de cette expérience est de simulée l'arrivée ou la reconnection d'une machine sur le switch.

Etat initial : Les machines 1 et 2 sont connectées alors que la machine 3 est déconnectée du switch. Les trois machines sont actives.

Déroulement : à un instant donné, je branche la machine 3.

Etat final : Les 3 machines sont actives et connectées au switch.

Trappes reçues :

Trappe n° 1 :

```
host=192.168.1.1,
port=8888,
community=public,
enterprise=1.3.6.1.4.1.9.1.429,
generic_trap=Link Up,
Varbinds=[1.3.6.1.2.1.2.2.1.1.5: 5,
1.3.6.1.2.1.2.2.1.2.5: FastEthernet0/5,
1.3.6.1.2.1.2.2.1.3.5: 6,
1.3.6.1.4.1.9.2.2.1.1.20.5: up, ]
```

Trappe n° 2

```
host=192.168.1.1,
port=8888,
community=public,
enterprise=1.3.6.1.4.1.9.9.41.2,
generic_trap=Enterprise Specific,
Varbinds=[1.3.6.1.4.1.9.9.41.1.2.3.1.2.15:LINK,
1.3.6.1.4.1.9.9.41.1.2.3.1.3.15: 4,
1.3.6.1.4.1.9.9.41.1.2.3.1.4.15: UPDOWN,
1.3.6.1.4.1.9.9.41.1.2.3.1.5.15: Interface FastEthernet0/5,
changed state to up,
1.3.6.1.4.1.9.9.41.1.2.3.1.6.15: 417338, ]
```

Observations :

La (re)connexion d'une machine sur le switch est également détectée, et comme dans le cas d'une déconnexion, nous recevons deux trappes.

4.8 Expérience n° 4 : Débranchement et reconnexion rapide

But : Le but de cette expérience est d'observer les réactions du système lors de déconnexion et reconnexion rapide d'un câble.

Etat initial : Les trois machines sont actives et connectées au switch.

Déroulement : à un instant donné, de débranche la machine 3 et je la rebranche aussitôt.

Etat final : Les trois machines sont à nouveau connectées au switch.

Trappes reçues :

Trappe n° 1 :

```
host=192.168.1.1,
port=8888,
community=public,
enterprise=1.3.6.1.4.1.9.1.429,
generic_trap=Link Up,
Varbinds=[1.3.6.1.2.1.2.2.1.1.5: 5,
1.3.6.1.2.1.2.2.1.2.5:FastEthernet0/5,
1.3.6.1.2.1.2.2.1.3.5: 6,
1.3.6.1.4.1.9.2.2.1.1.20.5: up, ]
```

Trappe n° 2

```
host=192.168.1.1,
port=8888,
community=public,
enterprise=1.3.6.1.4.1.9.9.41.2,
generic_trap=Enterprise Specific,
Varbinds=[1.3.6.1.4.1.9.9.41.1.2.3.1.2.15:LINK,
1.3.6.1.4.1.9.9.41.1.2.3.1.3.15:4,
1.3.6.1.4.1.9.9.41.1.2.3.1.4.15:UPDOWN,
1.3.6.1.4.1.9.9.41.1.2.3.1.5.15:Interface    FastEthernet0/5,
changed state to up,
1.3.6.1.4.1.9.9.41.1.2.3.1.6.15: 417338, ]
```

Observations :

Le comportement est exactement le même que lors de la connexion d'une machine. En fait, le switch ne donne pas l'indication du débranchement.

4.9 Expérience n° 5 : Reboot par redémarrage forcé:

But : Le but de cette expérience est d'observer si les résultats obtenus en débranchant et rebranchant rapidement un câble sont également obtenus lors du redémarrage d'une machine.

Etat initial : Les trois machines sont connectées au switch et actives.

Déroulement : à un instant donné, j'appui sur le bouton « reset » de la machine n° 3

Etat final : Les trois machines sont connectées au switch, et la machine 3 est cours de démarrage.

Trappe n° 1 :

```
host=192.168.1.1,
port=8888,
community=public,
enterprise=1.3.6.1.4.1.9.1.429,
generic_trap=Link Up,
Varbinds=[1.3.6.1.2.1.2.2.1.1.5:5,
1.3.6.1.2.1.2.2.1.2.5:FastEthernet0/5,
1.3.6.1.2.1.2.2.1.3.5: 6,
1.3.6.1.4.1.9.2.2.1.1.20.5: up, ]
```

Trappe n° 2

```
host=192.168.1.1,
port=8888,
community=public,
enterprise=1.3.6.1.4.1.9.9.41.2,
generic_trap=Enterprise Specific,
Varbinds=[1.3.6.1.4.1.9.9.41.1.2.3.1.2.15:LINK,
1.3.6.1.4.1.9.9.41.1.2.3.1.3.15:4,
1.3.6.1.4.1.9.9.41.1.2.3.1.4.15:UPDOWN,
1.3.6.1.4.1.9.9.41.1.2.3.1.5.15:Interface    FastEthernet0/5,
changed state to up,
1.3.6.1.4.1.9.9.41.1.2.3.1.6.15: 417338, ]
```

Observations :

On obtient exactement le même comportement que lors de l'expérience de déconnexion / reconnexion rapide. Un redémarrage forcé peut donc être identifié. J'ai également remarqué que le redémarrage logiciel d'une machine n'a aucune influence sur l'état de la carte réseau et par conséquent sur le switch.

4.10 Expérience n° 6 : Ecran bleu pendant un transfert de fichier

But : Le but de cette expérience est de simuler l'arrêt brutal d'une transmission réseau sans perte de connexion physique. L'idée est de simuler un « écran bleu », synonyme de blocage critique sous Windows pendant un transfert de fichier. La simulation d'un écran bleu est très simple sous Windows 98, puisqu'il suffit d'exécuter la ligne de commande :
c:\con\con. L'effet est immédiat et le système ne s'en remet pas.

Etat initial : Les trois machines sont actives et connectées au switch. Un transfert de fichier est amorcé entre la machine 1 et la machine 2

Déroulement : à un instant donné, je simule un écran bleu sur la machine 2.

Etat final : Les trois machines sont toujours connectées au switch, mais la machine 2 est dans un état instable et le transfert de fichier est interrompu.

Trappes reçues :

Dans ce cas je n'ai observé aucune trappe en provenance du switch.

Observations :

L'absence de trappes nous indique encore une fois que le switch ne semble pas analyser les données transitant par ses ports.

4.11 Expérience n° 7 : Arrêt / démarrage d'une machine.

But :

Cette expérience n'a de sens que pour valider mes attentes par rapport à ces événements. En effet, j'ai pensé que l'arrêt ou le démarrage d'une station produisait exactement le même comportement que le branchement ou le débranchement d'un câble. L'idée est relativement simple, dans un cas la machine est active et je l'arrête (brutalement ou non), dans l'autre la machine est éteinte et je la démarre.

Observation :

Selon mes attentes, le comportement est exactement le même que lors des manipulations avec un câble. En effet, bien que la cause de la panne soit différente, les conséquences au niveau de l'équipement sont exactement les mêmes.

4.11 Expérience n° 8 simulation d'un câble défectueux :

But : Le but de cette expérience est de simuler la présence d'un câble défectueux. C'est une panne que j'ai rencontrée fréquemment lors de l'installation de réseaux. La présence d'un tel câble ne se remarque par forcément, car dans certains cas les équipements réseau indiquent que le lien est disponible, alors qu'aucune opération réseau n'est possible.

Afin de réaliser cette expérience, j'ai conçu à l'aide d'un ami un câble dont je pouvais facilement couper et réactiver chaque brin. L'idée est de prendre un câble standard, et de relier chacun des brins par un interrupteur.
De cette manière, je peux couper en temps voulu chacun des brins du câble.

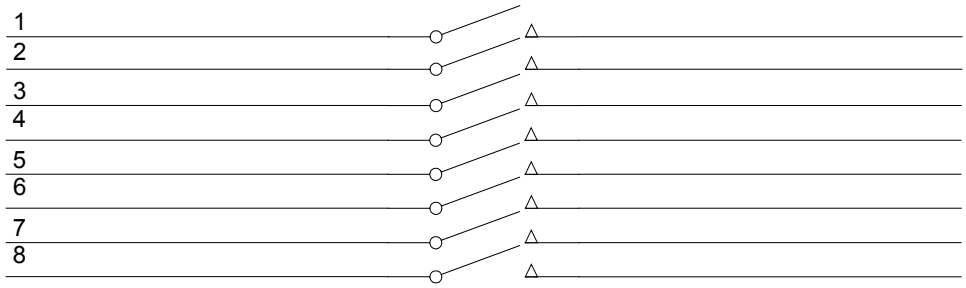

Figure 4.2 Schéma électrique du boîtier

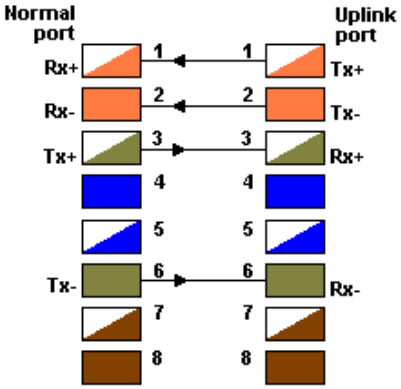

Figure 4.3 et 4.4 – Photo du boitier et schéma d'un câble Ethernet

Dans un câble Ethernet, nous savons que seul deux des quatre paires sont utilisées, c'est pourquoi je m'attendais à n'avoir aucune réaction dans 4 cas, à savoir les brins 4,5,7 et 8.

Etat initial :
Les trois machines sont actives et connectée au switch. La machine 2 est connectée au moyen du câble présenté ci-dessus. Un transfert de fichier est amorcé entre la machine 1 et la machine 2.

Déroulement :
Pendant le transfert de fichier, je vais couper un à un les brins 1,2,3 et 6 du câble connecté à la machine 2.

Etat final : Les trois machines sont actives et connectées au switch. Le transfert de fichier n'a pas abouti.

Déconnexion du brin n°1 : Le switch détecte une coupure de la même manière que lorsque l'on déconnecte complètement le câble.

Déconnexion du brin n°2 : Le résultat est identique à la déconnexion du premier brin.

Déconnexion du brin n°3 : Le transfert de fichier est interrompu. Par contre, le switch n'envoi aucune trappe. L'équipement n'a donc rien détecté d'anormal.

Déconnexion du brin n°6 : Le résultat est identique à la déconnexion du brin n° 3.

Observations

Cette fois encore nous avons la preuve que l'équipement n'effectue aucun contrôle sur l'état des communications. En effet, les brins 1 et 2 sont responsables de la transmission des données, tandis que les brins 3 et 6 sont responsables de la réception. Quand on coupe les 2 premiers brins, le switch détecte la disparition d'une tension. Par contre, le fait de couper les brins 3 et 6 n'influence en rien l'état électrique du port.

5. Discussion des résultats

5.1 Analyse des trappes.

Nous avons en fait quatre trappes différentes que l'on peu répartir en deux groupes. Les trappes spécifiques à CISCO et les trappes génériques. Dans ces deux groupes, nous avons une trappe indiquant le lien actif et une autre indiquant un lien inactif. Je vais examiner les informations que l'on trouve dans ces trappes.

Une trappe SNMP contient une certaine quantité d'informations, utiles ou non pour mon système de détection de panne. Tout d'abord, nous avons des informations sur la provenance de la trappe (entreprise, adresse, port), sur la communauté, et une quantité variable de données (Varbinds) indiquant des informations complémentaires. C'est surtout sur ces données que je me suis basé. Elles permettent entre autre de connaître la cause de la panne et le port concerné.

Je vais détailler les informations reçus dans les différentes trappes.

Informations générales :

host=192.168.1.1
Ceci nous indique que l'agent ayant envoyé la trap possède l'adresse IP 192.168.1.1.

port=8888:
Le port UDP sur lequel la trap a été reçue est le 8888.

community=public :
La communauté SNMP est « public »

Enterprise=1.3.6.1.4.1.9.9.41.2
Ceci nous indique la provenance de la trap, nous avons ici deux cas :

1.3.6.1.4.1.9.9.41.2, pour les traps CISCO

iso (1)
  org (3)
    dod (6)
      internet (1)
        private (4)
          enterprises (1)
            cisco (9)
              ciscoMgmt (9)
                ciscoSyslogMIB (41)
                  ciscoSyslogMIBNotificationPrefix (2)

Cette entrée représente le prefix que comporte toute entrée dans la MIB de management de CISCO. Nous pouvons l'observer dans les OIDs des trappes spécifique à CSICO

Entreprise=1.3.6.1.4.1.9.1.429

```
iso (1)
   org (3)
      dod (6)
         internet (1)
            private (4)
               enterprises (1)
                  cisco (9)
                     ciscoProducts (1)
                        catalyst295048G (429)
```

Cette fois, nous sommes dans le cas des trappes génériques. Il est précisé dans ce champ que l'agent qui a envoyé la trap est un produit CISCO du type Catalyst 2950.

Generic_trap=Enterprise Specific
Ce champ nous indique à quel type de trappe nous avons affaire. Dans le cas d'une trappe spécifique, nous aurons l'indication ci-dessus. Dans le cas des trappes génériques, nous aurons comme information le type de trap standard, en l'occurrence « Link_UP ou Link_DOWN.

Données complémentaires.

Les données complémentaires sont en fait une liste d'OID représentant des entrées de la MIB, ainsi que leur valeur. Ces entrées apportent différents types d'informations, et dépendent essentiellement de la nature de la trappe (spécifique ou générique).
Je vais décrire dans cette section les objets pour les deux trappes indiquant un « link up », en partant du principe que leurs pendants (link down) sont implicites. Pour m'aider dans l'analyse des trappes, j'ai utilisé un utilitaire en ligne proposé par CISCO [4].

Trappe générique link up

Liste des OIDs :

OID : 1.3.6.1.2.1.2.2.1.1.5
Type d'entrée : numérique
Valeur: 5

Position dans la MIB

```
iso (1)
   org (3)
      dod (6)
         internet (1)
            mgmt (2)
               mib-2 (1)
                  interfaces (2)
                     ifTable (2)
                        ifEntry (1)
                           ifIndex (1)
```

C'est objet fait référence à la liste des interfaces d'un équipement. Le dernier 5, indique qu'il s'agit de l'interface n°5, donc du port n° 5 du switch.

OID : 1.3.6.1.2.1.2.2.1.2.5
Type d'entrée: chaîne de caractères.
Valeur: FastEthernet0/5

Position dans la MIB

```
iso (1)
   org (3)
      dod (6)
         internet (1)
            mgmt (2)
               mib-2 (1)
                  interfaces (2)
                     ifTable (2)
                        ifEntry (1)
                           ifDescr (2)
```

C'est objet est en fait le pendant textuel du premier objet présenté.

OID : 1.3.6.1.2.1.2.2.1.3.5
Type d'entrée: numérique
Valeur: 6

Position dans la MIB

```
iso (1)
   org (3)
      dod (6)
         internet (1)
            mgmt (2)
               mib-2  (1)
                  interfaces (2)
                     ifTable (2)
                        ifEntry (1)
                           ifType (3)
```

Cette entrée nous indique le type d'interface concerné. Le type d'interface est représenté par une constante numérique, dans notre cas 6. Selon la documentation proposée par CISCO, cette constante représente le type ***« ethernetCsmacd »***, ce qui nous ne surprend pas dans la mesure où l'on travail avec Ethernet.

OID : 1.3.6.1.4.1.9.2.2.1.1.20.5
Type d'entrée: Chaîne de caractères
Valeur: up

Position dans la MIB

```
iso (1)
   org (3)
      dod (6)
         internet (1)
            private (4)
               enterprises (1)
                  cisco (9)
                     local (2)
                        linterfaces (2)
                           lifTable (1)
                              lifEntry (1)
                                 locIfReason (20)
```

Cette entrée spécifique à la MIB CISCO nous informe sur la raison de la trap, dans notre cas l'interface concernée est devenue active. Cette entrée n'est pas indispensable dans ce cas, car nous savons déjà par le type de trappe (LINK_UP) la raison qui l'a déclenchée.

Trap spécifique CISCO link UP

OID : 1.3.6.1.4.1.9.9.41.1.2.3.1.2.15
Type d'entrée: Chaîne de caractères
Valeur: LINK

Position dans la MIB

```
iso (1)
  org (3)
    dod (6)
      internet (1)
        private (4)
          enterprises (1)
            cisco (9)
              ciscoMgmt (9)
                ciscoSyslogMIB (41)
                  ciscoSyslogMIBObjects (1)
                    clogHistory (2)
                      clogHistoryTable (3)
                        clogHistoryEntry (1)
                          clogHistFacility (2)
```

Cette entrée de la MIB nous informe sur le service concerné par la trappe. Dans notre cas il s'agit du lien. Le chiffre 15 présent à la fin de l'OID est en fait un numéro de trappe, et est le même pour tous les OIDs présent dans celle-ci. A chaque trappe, celui-ci est incrémenté de 1.

OID : 1.3.6.1.4.1.9.9.41.1.2.3.1.2.15
Type d'entrée: Numérique
Valeur: 4

Position dans la MIB

```
iso (1)
  org (3)
    dod (6)
      internet (1)
        private (4)
          enterprises (1)
            cisco (9)
              ciscoMgmt (9)
                ciscoSyslogMIB (41)
                  ciscoSyslogMIBObjects (1)
                    clogHistory (2)
                      clogHistoryTable (3)
                        clogHistoryEntry (1)
                          clogHistSeverity (3)
```

Cette entrée nous informe sur le niveau de gravité de la panne. Dans notre cas il vaut 4. Ceci correspond au niveau « erreur »

Voici la liste des niveaux d'erreur que CISCO propose.

1:emergency
2:alert
3:critical
4:error
5:warning
6:notice
7:info
8:debug

OID : 1.3.6.1.4.1.9.9.41.1.2.3.1.4.15
Type d'entrée: Chaîne de caractères
Valeur: UPDOWN

Position dans la MIB

1.3.6.1.4.1.9.9.41.1.2.3.1.4.15: UPDOWN,

iso (1)
org (3)
dod (6)
internet (1)
private (4)
enterprises (1)
cisco (9)
ciscoMgmt (9)
ciscoSyslogMIB (41)
ciscoSyslogMIBObjects (1)
clogHistory (2)
clogHistoryTable (3)
clogHistoryEntry (1)
clogHistMsgName (4)

Cette entrée est prévue pour indiquer la catégorie de l'événement qui a généré la trappe. Dans notre cas, il s'agit d'un passage de l'état inactif à l'état actif.

OID : 1.3.6.1.4.1.9.9.41.1.2.3.1.5.15
Type d'entrée: Chaîne de caractères
Valeur: Interface FastEthernet0/5, changed state to up

Position dans la MIB

```
iso (1)
   org (3)
      dod (6)
         internet (1)
            private (4)
               enterprises (1)
                  cisco (9)
                     ciscoMgmt (9)
                        ciscoSyslogMIB (41)
                           ciscoSyslogMIBObjects (1)
                              clogHistory (2)
                                 clogHistoryTable (3)
                                    clogHistoryEntry (1)
                                       clogHistMsgText (5)
```

Cette entrée nous donne une description détaillée de l'événement à l'origine de la trap.

OID : 1.3.6.1.4.1.9.9.41.1.2.3.1.6.15
Type d'entrée : Numérique
Valeur: 417338

Position dans la MIB

1.3.6.1.4.1.9.9.41.1.2.3.1.4.15: UPDOWN,

```
iso (1)
   org (3)
      dod (6)
         internet (1)
            private (4)
               enterprises (1)
                  cisco (9)
                     ciscoMgmt (9)
                        ciscoSyslogMIB (41)
                           ciscoSyslogMIBObjects (1)
                              clogHistory (2)
                                 clogHistoryTable (3)
                                    clogHistoryEntry (1)
                                       clogHistTimestamp (4)
```

Cette entrée correspond au « up time » du système. Ceci nous donne le temps de fonctionnement depuis de démarrage du switch jusqu'à l'émission de la trappe en millisecondes.

Nous venons de voir qu'une trappe SNMP peut transporter un nombre non négligeable d'information. D'autre part, nous pouvons constater que les trappes génériques et les trappes spécifiques apportent dans l'ensemble les mêmes informations.

5.2 Temps de réaction

Un des points qui n'a pas encore été abordé est le temps de réponse ou de réaction de l'équipement. C'est peut-être ici que se situe le point faible de ce système.
En effet, après le débranchement d'un câble, il faut attendre environ 2 secondes avant de recevoir la trappe correspondante. Dans un premier temps, j'ai pensé qu'il s'agissait tout simplement du temps de réaction du système. Cependant, mes observations sur les branchements / débranchements rapides me suggère plutôt une autre explication.

La figure 5.1 est un diagramme de temps mettant en évidence dans un premier cas la détection d'un débranchement, et ensuite la réaction du système lors d'une déconnexion / reconnexion rapide. On voit bien que si le temps écoulé entre ces deux actions est inférieur au facteur t, la déconnexion ne sera pas détectée. C'est en fait ce facteur qui permet de faire la différence entre un arrêt et un redémarrage.

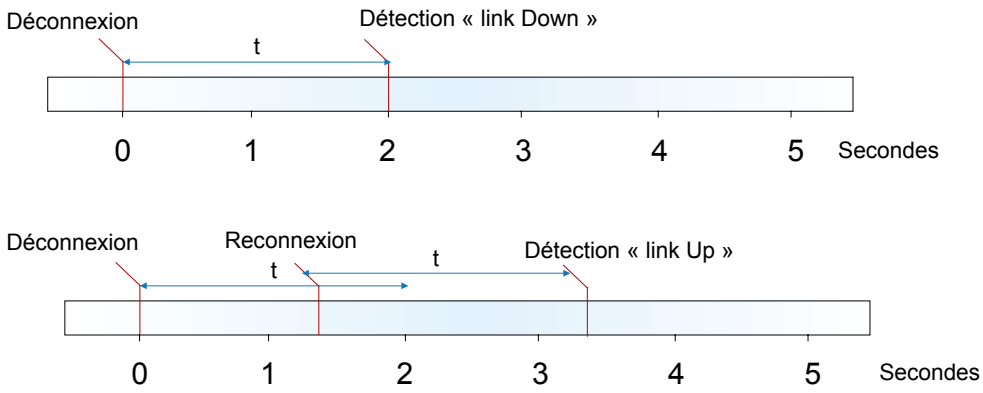

Figure 5.1 – Diagramme de temps de détection

Si le système indiquait tout de suite la chute du lien, il ne serait plus possible de faire la différence entre un redémarrage et un démarrage normal. De plus, ce temps de latence assure qu'il s'agit réellement d'une panne, et non d'une simple perturbation électrique.

Dans mon application, je pense que ce long temps de réaction n'est pas obligatoirement un handicap, dans la mesure où l'on a la garantie que l'information obtenue est fiable (ce n'est pas une suspicion comme dans la

détection proposée par Fabien Reichenbach. D'autre part, la présence de ce temps de latence nous permet d'effectuer des détections plus ciblées comme le redémarrage forcé d'une machine.

5.3 Détections pouvant être effectuées

Comme je l'ai déjà mis en avant, l'équipement que j'avais à disposition ne détecte que les pannes qui se manifestent à un niveau électrique. En effet, toutes mes tentatives pour amener l'équipement à rencontrer des anomalies au niveau des transmissions sont restées veines.
Cependant, les détections faites au niveau électrique sont réalisées de manière efficace et précise. Au aucun moment je n'ai observé d'erreur de diagnostique. De plus, les informations communiquées dans les trappes envoyées sont claires et précises.

Le tableau suivant résume les pannes pouvant être détectées ainsi que leur manifestation au sein de l'équipement :

Panne / événement	Manifestation
Coupure de l'alimentation d'une machine	Perte du lien
Arrêt d'une machine	Perte du lien
Déconnexion du câble	Perte du lien
Démarrage d'une machine	Le lien devient actif
Redémarrage brutal d'une machine	Le lient devient actif avant que la perte du lien ne soit détectée
Branchement d'une machine	Le lien devient actif

Figure 5.2 – Résumé des type de pannes et de leur manifestation

On peut par exemple remarquer que la détection du branchement / démarrage d'une machine peut alerter l'administrateur réseau de l'arrivée ou de la présence d'une machine non attendue sur le parc.

Nous pouvons donc remarquer que ce type d'appareil, même s'il ne traite que le niveau électrique, nous permet déjà d'obtenir un nombre intéressant d'informations et d'événement pouvant intervenir dans un réseau.

L'avantage principal de cette méthode est que chaque station d'un réseau peut maintenir en temps réel l'état du parc au complet.

5.4 Fiabilités des résultats

Un des points les plus importants que je souhaite souligner est la fiabilité des détections effectuées par l'équipement. En effet, les détections réalisées par ce type d'équipement sont « sûr » dans le sens ou ce n'est pas une suspicion de panne. Si l'appareil détecte qu'un lien est tombé, c'est qu'il ne parvient plus à communiquer au niveau électrique.

Cette fiabilité est un point très important pour moi, car il permet de lever les doutes qui règnent lors d'une détection de panne par attente de trappe.

Un autre point que je souhaite également mettre en avant est que le switch reste l'unique élément de détection, et ne dépend pas de l'état des autres équipements pour fonctionner. Donc à moins que lui-même ne cesse de fonctionner, ce qui compromet bien évidemment le fonctionnement de tout le réseau, les informations de détection de panne seront toujours disponibles pour toutes les machines en état de fonctionnement.

6 Application Java

6.1 Framework de Fabien Reichenbach

Pour réaliser mon application de détection de panne, je me suis essentiellement basé sur l'implémentation de Fabien Reichenbach. J'ai repris les Framework SNMP qu'il a utilisé en me basant essentiellement sur son implémentation du moniteur.

6.2 Mon implémentation

Le but principal de mon application est d'afficher sur chaque station l'état complet du réseau. La topologie du réseau est décrite dans un fichier de configuration que je détaillerai dans la section suivante.

Diagramme de classe :

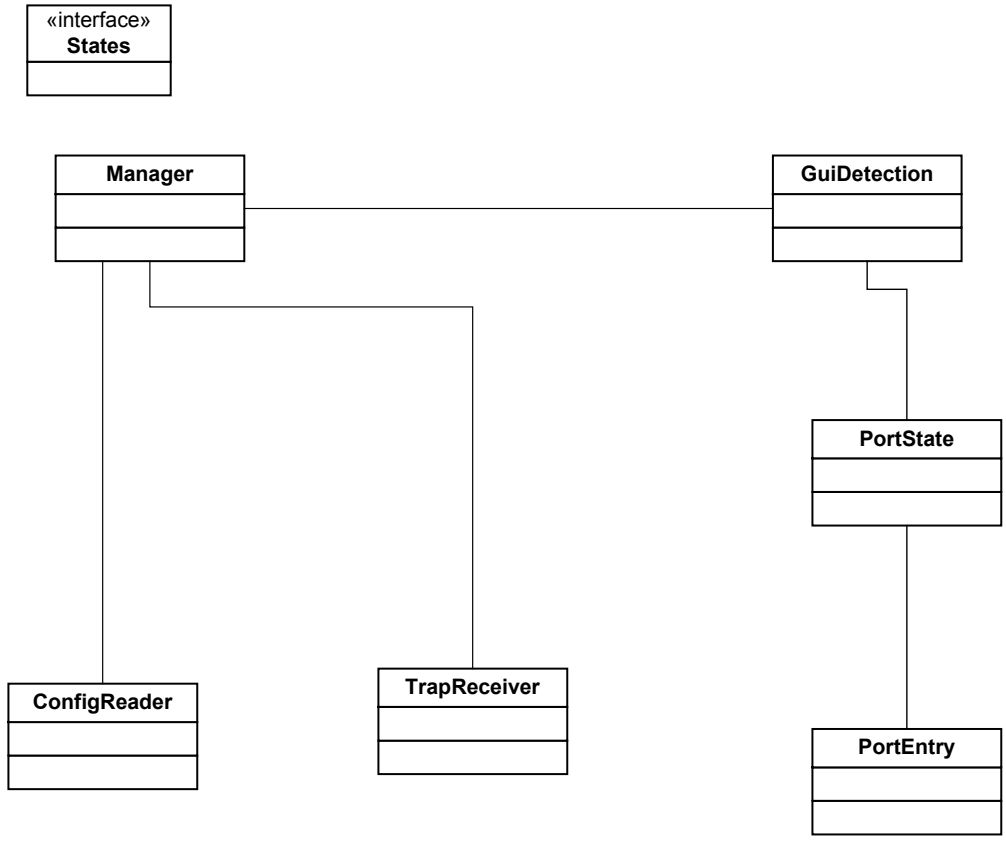

Figure 6.1 – Diagramme de classe

Description des classes :

Interface States
Cette interface sert simplement à définir les états des ports, UP ou DOWN.

Manager
Cette classe est la classe de base de l'application. Elle gère l'initialisation et la communication avec l'interface graphique.

GuiDetection
Cette classe gère l'affichage de la topologie du réseau.

PortState et PortEntry
Ces classes s'occupent de la représentation graphique des stations et du calcul de leur état.

ConfigReader
Cette classe sert à lire et analyser le fichier de configuration

TrapReceiver
Cette classe s'occupe de toute la gestion des trappes SNMP.

Fonctionnement du système :

La classe de base est la classe Manager. Au lancement de l'application celle-ci va dans un premier temps construire la topologie fournie dans le fichier de configuration au moyen de la classe ConfigReader. Une fois la topologie connue, le manager va créer autant d'objet PortState que de stations présente dans la topologie, et les ajouter à la classe GuiDectector.
Ensuite, le manager va initialiser le récepteur de trappes en lui précisant l'adresse IP de l'agent. Le port utilisé est le 8888.
Lorsqu'une trappe est détectée par le récepteur de trappe, celui-ci va la décoder et la transmettre au manager. Ce dernier va déterminer de quel type de trappe il s'agit, et va transmettre l'information à la classe GuiDetector. Cette dernière va avertir chaque objet PortState qu'elle contient.

Une fois qu'un objet PortState reçoit une notification de trappe, celui-ci va mettre à jour sa représentation graphique en ayant au préalable calculer son état et la cause du changement d'état.

Analyse des trappes :

Pour l'analyse des trappes, j'ai utilisé les deux trappes standard envoyées par le switch de manière à créer une application la plus compatible possible.

Je me base d'une part sur l'entrée de la MIB qui définit le port concerné, et d'autre part sur l'indication de l'état du lien ; UP ou DOWN.

6.3 Utilisation de mon application

Pour lancer mon application, il suffit d'exécuter le fichier JAR en lui précisant l'adresse IP de l'agent comme paramètre :

```
java –jar Dectecteur.jar 192.168.1.1
```

Il est nécessaire que les jars suivants se trouvent dans le répertoire d'exécution :

```
AdventNetSnmp.jar
AdventNetSnmpAgent.jar
crimson.jar
jaxp.jar
xalan.jar
```

Ce sont les bibliothèques nécessaires au fonctionnement du Framework de Fabien Reichenbach, et par conséquent nécessaires à mon application.

Le fichier de configuration :

Le fichier doit s'appeler « config.txt » et se trouvé dans le répertoire d'exécution de l'application. La description de la topologie est très simple, puisque chaque ligne représente une station. Pour chaque station, il faut préciser son nom suivit de « : » suivit du port du switch sur lequel elle est connectée « Port 1, Port 2, … ». Le fichier doit se terminer par une ligne vide.

Voici un exemple de configuration :

```
Portable:Port 1
Machine1:Port 3
Machine2:Port 5
```

7 Applications

7.1 Type de panne et contexte

Les types de pannes pouvant être détectés ont déjà été discutés dans la partie « résultats » de ce document. Je veux insister ici sur le fait que les pannes détectées par ce système sont des pannes tout à fait courantes dans un réseau réel. De plus, le fait que chaque station du réseau reçoive et interprète les informations venant de l'agent SNMP nous permet d'envisager des systèmes de résistance aux pannes très fiable.

Nous avons selon moi deux applications très intéressantes de ce type de système de détection de panne.

D'une part, nous avons les systèmes de pure détection de panne. Une des applications la plus importante est dans la gestion de systèmes répartis. L'avantage principal dans ce genre de situations est que chaque machine est en tout temps informée de l'état du réseau et ce de manière extrêmement fiable. Ceci nous permet de supprimer la suspicion, et donc de réagir de manière beaucoup plus rapide et sans risque de mauvaise détection.

D'autre part, nous pouvons réaliser une application de surveillance du réseau. De cette manière, nous pouvons connaître à tout moment les stations en état de fonctionnement, où savoir si de nouvelles stations ont été ajoutées. Ce type d'application peut être très intéressant pour des administrateurs réseau gérant des parcs d'une certaine taille

7.2 Utilisation de système software en complément

Comme je l'ai déjà précisé, le système de détection que j'ai réalisé ne prend en compte que des pannes détectables électriquement. Je pense que pour des applications nécessitant la détection de panne également au niveau software, il serait relativement intéressant d'utiliser ce type de système en compléments à des ceux fonctionnant par attente de trappe.

7.3 Gamme et prix des équipements compatibles.

Selon les informations communiquées par CISCO, tous les modèles implémentant le protocole SNMP sont basés sur la même implémentation. Donc on peut partir du principe que tout modèle implémentant SNMP peut convenir à de telles applications. De plus, je me suis renseigné sur les trappes standard, et il s'avère que la détection des coupures de lien fait partie des fonctionnalités les plus souvent implémentées.

On peut donc penser que n'importe quel équipement implémentant SNMP peut être utilisé pour de tels systèmes de détection. Il faudra cependant certainement adapter le logiciel de réception des trappes en fonction des particularités des constructeurs. Toutefois, ces modifications reste simples à effectuer et on peut espérer que la plupart des constructeurs se seront basés sur les trappes standard.

Gammes de prix chez CISCO :

L'appareil que j'avais à disposition coûte environ 5000 francs. Il faut noter toutefois qu'il s'agit d'un modèle 48 ports. Dans la même gamme en 12 ports, on trouve des modèles à moins de 1000 francs.

Chez d'autres marques on peut également trouver des équipements similaires à moins de 1000 francs, notamment chez NETGEAR avec un modèle 24 ports supportant SNMP pour moins de 900 francs.

8 Conclusions

8.1 Conclusions

La première conclusion que je peux tirer de ce projet est qu'il est effectivement possible de réaliser de la détection de panne sur un réseau avec un équipement implémentant SNMP. De plus, ce type d'équipement bien que plus cher que les produits grand publique, peut tout à fait rentrer dans des budgets dimensionnés en fonction de la taille du réseau à surveiller. Les faiblesses que je peux relever se situent essentiellement sur le temps de détection passablement long, mais qui reste toutefois raisonnable. Cependant, comme je l'ai déjà dit, ce handicap est en bonne partie compensé par les possibilités de détections supplémentaires offertes.

Concernant les types de pannes détectées, je reste quand même étonné que ce type d'équipement ne réalise aucune analyse au niveau des données transmises. Cependant, il est vrai que de telles analyses demanderait des compromis assez importants sur les performances des équipements. Malgré le nombre restreint de pannes détectées, celles-ci sont tellement explicites et précises, que leur interprétation permet de réaliser un contrôle passablement étendu sur un réseau. Je pense que la fiabilité d'un tel système permet d'envisager des applications très performantes et très efficaces, même pour des réseaux de grande taille. De plus, ce système implique un trafic supplémentaire totalement négligeable, et ne demande que peux de moyens, hormis le matériel, pour être mis en œuvre.

En ce qui concerne les améliorations possibles, je pense qu'il serait intéressant d'utiliser le mode d'interrogation du protocole SNMP, notamment pour réaliser de la découverte de topologie. Ceci apporte deux avantages conséquents au système. D'une part, il n'est plus nécessaire d'indiquer la configuration du réseau, opération qui peut s'avérer extrêmement lourde pour des parcs de grande taille. D'autre part, le système tel que je l'ai implémenté ne gère pas la cohérence de l'état du réseau pour une machine qui aurait rencontré un problème (par exemple un redémarrage) et qui est à nouveau fonctionnel. En effet, les notifications de pannes étant transmises par des trappes, si une station est hors service lorsqu'un problème se produit, elle ne sera pas avertie. Au moment où elle est à nouveau active, elle n'a aucun moyen de connaître les événements qui sont intervenus pendant sa période d'inactivité. En interrogeant l'équipement, elle peut par contre déterminer l'état complet du réseau, et ainsi le système de détection peut à nouveau être opérationnelle.

Pour réaliser un système complet de détection de panne, je pense que la combinaison d'un système de détection assisté d'un équipement réseau avec

le système proposé par Fabien Reichenbach est une solution efficace pour combler les faiblesses respectives des deux concepts. D'une part le système que je propose est fiable mais détecte un nombre limité de panne, d'autre part celui de Fabien Reichenbach permet de détecter un très grand nombre de pannes, mais souffre d'une grande incertitude. En intégrant ces deux systèmes, on dispose d'une part de leurs avantages, et d'autre part, leur complémentarité nous permet d'établir des diagnostiques passablement avancé lors d'une panne.

Ce projet m'a apporté énormément de connaissances tant au niveau du protocole SNMP qu'au niveau des équipements réseau. J'ai pu comprendre en détails les raisons de l'élaboration de ce protocole et me faire une bonne idée sur la quantité d'application qu'il permet de réaliser. Ce projet m'a également permis de mieux comprendre le fonctionnement d'un réseau, et la complexité que représente des applications comme la détection de panne.

8.2 Remerciements

Service informatique central de l'EPFL et notamment Gabriella Quiblier, Vito De Marinis, Nathalie Rumley, Christiane Dubrit et Josiane Scalfo, pour le prêt du matériel réseau.

Matthias Wiesamm pour son soutien et ses conseils tout au long du projet.

Jean-Claude Baumann, pour son aide à la confection de matériel de test.

9. Bibliographie

[1] J. Case, M. Fedor, M. Schoffstall and J. Davin. A simple network management protocol. (SNMP). RFC 1157, Interbet Engineering Task Force (IETF), 1990.

[2] F. Reichenbach. Service SNMP de détection de faute pour des systèmes répartis. Projet de diplôme, école polytechnique fédérale de Lausanne, février 2002.

[3] CISCO – SNMP. http://www.cisco.com/warp/public/535/3.html. Description de l'implémentation SNMP de CISCO.

[4] Cisco SNMP Object Navigator. http://www.cisco.com/pcgi-bin/Support/Mibbrowser/unity.pl?tab=2. Utilitaire d'analyse d'Oid SNMP.